加微信：642945106 发送“ 赠送 ”领取赠送精品课程
发数字“ 2 ”获取众筹列表

06 | 性能数据的分析：如何从大量数据中看出想要的信号？

2019-12-09 庄振运

讲述：庄振运
时长 13:22 大小 12.26M

你好，我是庄振运。

我们这一讲来谈谈如何分析我们所得到的性能数据。现代的应用程序和互联网服务系统都比较复杂，要关心的性能参数也很多，所以你从各种渠道得到的性能相关的数据量往往很大。那么要如何从大量的数据中找出我们所关心的特征和规律呢？这就需要你对数据做各种分析和对比了。

这一讲要解决的核心问题就是：**如何从大量数据中看出想看的信号？**

当人沉浸在大量数据中时，是很容易迷失的。而“不识庐山真面目”的原因，当然是“只缘身在此山中”了。但这不能作为借口，我们需要练就“慧眼识珠”的本领，做到对各种性能

数据一目了然，才能够做出一针见血的分析。

为了帮助你练就这样的本领，今天我们首先讲一下常见的算法复杂度和性能分析的目的，然后针对一个性能指标来分析，再延伸到对多个性能指标进行对比分析，最后谈谈进行数据分析的几个教训和注意点。

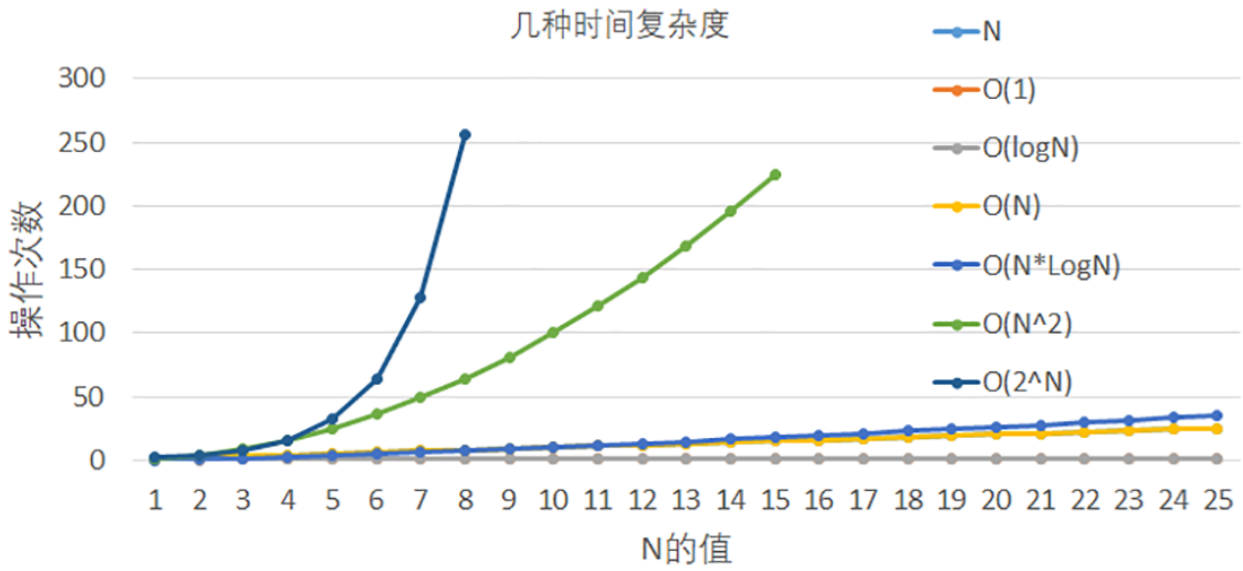
算法的时间复杂度

先简单地聊一下算法的时间复杂度（Time Complexity）。复杂度一般表示为一个函数，来定性描述该算法的期待运行时间，常用大 O 符号表述。

考虑程序和算法的时间复杂度时，大家通常关注的是某个解决方案属于哪个时间复杂度。具体来讲，有六种复杂度是比较普遍的，这里按照从快到慢的次序依次介绍：

1. 常数时间，O(1)：判断一个数字是奇数还是偶数。
2. 对数时间，O(Log(N))：你很熟悉的对排序数组的二分查找。
3. 线性时间，O(N)：对一个无序数组的搜索来查找某个值。
4. 线性对数时间，O(N Log(N))：最快的排序算法，比如希尔排序，就是这个复杂度。
5. 二次时间，O(N^2)：最直观也最慢的排序算法，比如冒泡，就是这个复杂度。
6. 指数时间，O(2^N)： 比如使用动态规划解决旅行推销员问题。这种复杂度的解决方案一般不好。

把这几个算法复杂度放在一张图中表示出来，你可以清楚地看出它们的增长速度。



大体上来讲，前四种算法复杂度比较合理，而后面两种（也就是 N 平方和指数时间）就不太能接受了，因为在数据量大的时候，运行时间很快就超标了。

防止断更 请务必加
首发微信：1716143665

性能数据分析的目的

说完常见的算法复杂度，我们再来看看做性能数据分析的目的是什么。

性能相关的数据有很多种，比如系统和模块的运行时间、客户访问延迟、客户滞留时间、服务吞吐率、程序的 CPU 占用时间等等。最常见的性能数据就是客户的访问延迟、吞吐量和运行时间。比如，我们如果去分析某个应用程序或者代码模块的运行时间，往往就会立刻暴露其相应的性能。

当收集到性能数据以后，我们首先需要判断它们的值到底是正常还是不正常，这就需要经验和知识才能判断。比如一个简单的数据库服务查询，一般端到端的延迟也就几百毫秒。如果这个值是 10 秒钟，那么就不太正常。发现不正常的数据，一般就需要做更多的分析和测试来发现原因，比如是网络延迟，数据库的服务延迟，抑或是其他问题。

再举一个例子，如果我们是在测量硬盘的访问时间，如果发现这个指标超过 1 秒，那么很可能磁盘这边出异常了。

如果观测到针对某个指标的一系列性能数据，那就需要判断这个指标有没有随着时间或者其他变量的变化而变差（Regression）和变好（Improvement），然后需要根据这个判断，进行进一步的分析及采取措施。

我们还经常需要对某个性能指标做预测。这种情况就需要研究数据的趋势（Trend）。根据情况做些预测分析，比如根据这个指标的历史数据来进行曲线拟合。

有时候某个性能指标会出现问题，比如如果应用程序有内存消耗和内存泄漏，那么我们就需要测量更多的相关指标来做深层的分析，来发现到底是哪个模块，哪行代码导致了内存泄漏。

我们也经常需要把几个性能指标的数据联系起来一起分析。比如发现系统的性能瓶颈在哪里？是在 CPU 的使用上、网络上，还是存储的 IO 读写上等等。如果应用程序和系统性能出了问题，那么考察多个性能指标的数据和它们的关系，可以帮我们做根因分析，发现真正的问题所在。

对一个时间序列的分析

一般来讲，一个性能指标，按照时间顺序得到的观测值，可以看作是一个时间序列。很多分析就是针对这个时间序列进行的。

首先需要指出的是，性能数据的时间序列往往不是均匀平滑的，反而会有各种有规律的峰值。比如，客户对一个网站的访问量是随着时间变化而变化的。每天 24 小时都不同，对多数互联网系统来讲，白天上班时间的访问量比较大；每周 7 天里面，工作日流量比较大；而节假日（比如新年）的流量又和其他时间不同。所以，需要你根据具体的情况来决定要不要做特殊的考虑。

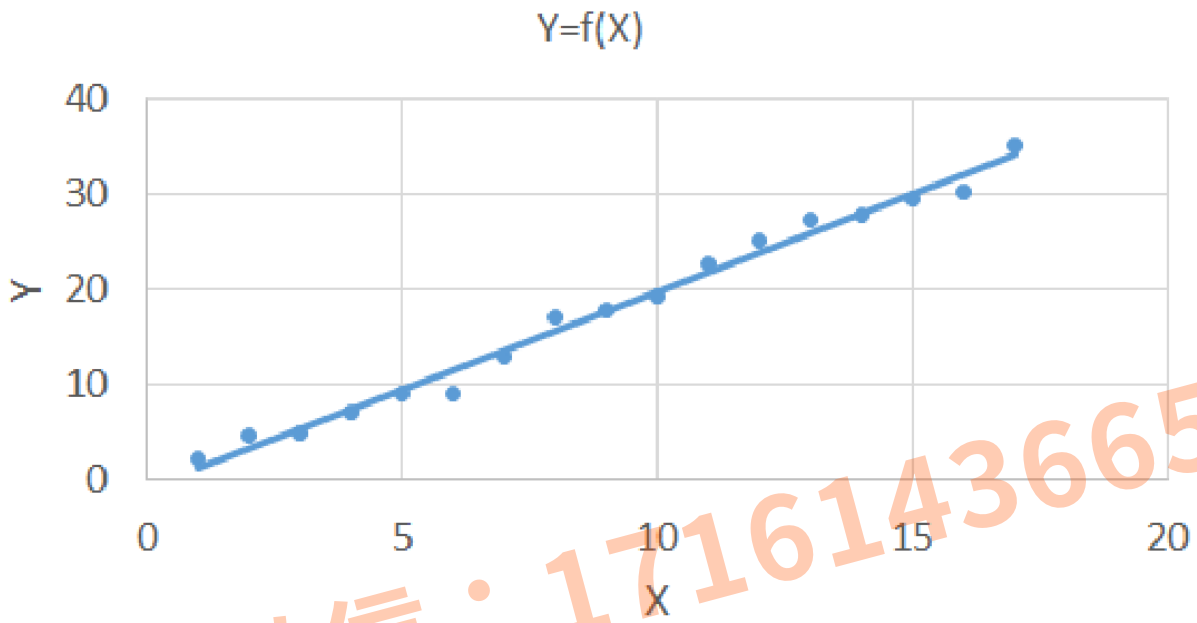
再有一点，就是对网站而言，客户响应时间往往需要考虑百分位的数字，比如 P90、P95、P99，甚至 P99.9 的用户响应时间。因为这些数字可以保证一个系统的响应时间是不是满足了绝大多数用户的要求。

那么，针对一个性能数据的时间序列，我们要如何看数据的规律和趋势呢？

经常使用的方法是进行**线性回归分析**（Linear Regression）。线性回归是通过拟合自变量与因变量之间最佳线性关系，来预测目标变量的方法。线性回归往往可以预测未来的数据

点。比如根据过去几年的每月消费支出数据，来预测明年的每月支出是多少。

注意所谓的“最佳”线性关系，是指在给定形状的情况下，没有其他位置会产生更少的误差。如下图所示，以平面点为例，如果有 N 个样本点，线性回归算法就是求一条直线 Y=f(X)。使得各点到这个曲线的距离的绝对值之和最小。



拼课微信：17161436665

除了研究数据的趋势和未来预测，还有几种重要的分析，比如**分类**（Classification）、**聚类**（Clustering）以及**决策树**（Decision Tree）。分类是将类别分配给数据集合，帮助更准确地预测和分析。聚类是把相似的东西分到一组。决策树也叫分类树或回归树，每个叶节点存放一个类别，每个非叶节点表示一个特征属性上的测试。

对不同时间序列的分析

在一大堆性能数据面前，经常需要比较各个性能指标的时间序列来确定一个系统和服务的瓶颈，也就是最制约系统性能扩展的资源。

在多数情况下，瓶颈资源是常用的几种，比如 CPU、网络、内存和存储。但是有些情况下其他不太常见的资源也可能成为瓶颈，比如转换检测缓冲区 TLB（Translation Lookaside Buffer）（这个我们以后会讲到）。

如果几个时间序列在时间上是一致的，但是对应不同的性能指标，比如一个是 CPU 的使用率，另一个是吞吐率。我们有时候需要研究时间序列的相关性（Correlation of time series），从中可以得出很多有用的观察推断。

数据的相关性是指数据之间存在某种关系，可以是正相关，也可以是负相关。两个数据之间有很多种不同的相关关系。比如，我们经常需要计算两个随机矢量 X 和 Y 之间的协方差 cov（X, Y）（Covariance) 来衡量它们之间是正相关还是负相关，以及它的具体相关度。

数据分析的教训和陷阱

性能数据的分析并不容易，一不小心就会落入各种陷阱或者踩到坑。下面举几个需要特别注意的方面。

第一是数据的相关和因果关系。

有时候几个时间序列之间可以很清楚地看出有很强的相关性质，但是对它们之间的因果关系却不能判定。换句话说，通过单纯的数据分析可以证实数据的相关性，但是还需要其他知识才能更准确地判断谁是因、谁是果。这一点我们必须非常清楚，因为在很多性能问题讨论和根因分析的场合，我们非常容易武断地犯这样的错误，而导致走弯路。

更复杂的情况是有时候系统性能变坏，是因为几个指标互为因果，或者构成**环形因果**，也就是互相推波助澜。实际分析起来非常有挑战性，这就需要我们对整个系统和各个性能指标了如指掌。

第二是数据的大小和趋势。

面对性能相关的数据并判断它们是“好”还是“坏”是很难的。经常听到有人问一个问题：

“客户平均访问逗留时间多长比较好？”

这个问题没有一个简单的答案。这取决于每个网站的特性，以及我们想要实现的目标。对一个网站而言很正常的逗留时间，可能对另一个网站而言非常糟糕。在很多情况下，比单纯数字大小更重要的是数据的趋势，比如某个时期是上升还是下降，变化的幅度有多大等等。

第三是数据干净与否。

如果数据集合来自多个数据源，或者来自复杂的测试环境，我们需要特别注意这些数据里面有没有无效数据。如果不能剔除无效数据，那么整体数据就“不干净”，由此而得出的结论经常会“失之毫厘，谬以千里”。

第四是对性能数据内在关系的理解。

性能数据分析的核心，就是要理解各个性能指标的关系，并且根据数据的变化来推断得出各种结论，比如故障判别、根因分析。如果简单地把性能数据当作普通的时间序列来分析，那就往往没有抓住精髓。举个简单例子，Linux 系统的空闲内存其实就是一条时间序列，它或许显示快到 0 了，看起来性能问题出在这里。但是稍微了解 Linux 系统内存管理知识的人，就知道这个指标非常不可靠。

总结

性能工程和优化离不开对大量性能数据的研究和分析，那么如何“拨开云雾见天日”，看出里面的端倪和问题呢？我们这一讲就讨论了几种情况，包括对一个和多个时间序列的分析；也讨论进行数据分析的时候需要注意的地方。

古人讲“格物致知”；对我们来讲，性能数据就是我们要“格”的物。只有合理而系统地分析这些数据，才能收获“守得云开见月明”的恍然大悟之感。

思考题

对数据的统计分析和处理需要遵循科学的方法，否则，如果处理不当，根据数据得出的结论会严重误导你。回想一下过去的工作中，有没有这样的例子？从这些例子中有没有学到教训?

欢迎你在留言区分享自己的思考，与我和其他同学一起讨论，也欢迎你把文章分享给自己的朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　05 | 概率统计和排队论：做性能工作必须懂的数理基础

精选留言 (1)

写留言

Q

2019-12-09

做运维的免不了要经常和性能数据打交道，往往很多时候，不细致了解数据内在含义，仅靠字面意思，往往很容易走入误区。比如CPU IO Waite，当这个值升高的时候，大多数人，包括我自己，会认为系统IO出现瓶颈问题了。直到我阅读了一篇关于CPU IO Waite的文章之后，我才明白，这个值不一定说明当前系统IO出现问题。简单来说，能导致CPU IO Waite值升高，有两个条件:...

展开

1